AMENDMENT AND RESPONSE UNDER 37 CFR § 1.116 – EXPEDITED PROCEDURE
Serial Number: 10/643,741
Filing Date: August 18, 2003
Title: MULTISTREAM PROCESSING MEMORY-AND BARRIER-SYNCHRONIZATION METHOD AND APPARATUS
Page 2
Dkt: 1376.733US1

IN THE CLAIMS

Please amend the claims as follows.

1.   (Currently Amended) An apparatus comprising:

a memory interface;

a plurality of queues connected to the memory interface, including a first queue and a second queue, wherein each of the ~~first~~ plurality of queues ~~for holding~~ holds ~~a plurality of~~ pending memory requests and enforces an ordering in the commitment of the pending memory requests to memory;

one or more instruction-processing circuits, wherein each instruction-processing circuit is operatively coupled through the plurality of queues to the memory interface and wherein each of the plurality of instruction-processing circuits inserts one or more memory requests into at least one of the queues based on a first memory operation instruction, inserts a first synchronization marker into the first queue and inserts a second synchronization marker into the second queue based on a synchronization operation instruction, and inserts one or more memory requests into at least one of the queues based on a second memory operation instruction; and

a first synchronization circuit, operatively coupled to the ~~first~~ plurality of queues, that selectively halts processing of further memory requests from the first queue based on the first synchronization marker reaching a predetermined point in the first queue until the corresponding second synchronization marker reaches a predetermined point in the second queue;

wherein each of the memory requests is a memory reference, wherein the memory reference is generated as a result of execution of instructions by the instruction-processing circuits.

2.   (Previously Presented) The apparatus of claim 1, wherein the first queue is used for only synchronization markers and vector memory references, and the second queue is used for only synchronization markers and scalar memory references.

3.   (Previously Presented) The apparatus of claim 2, wherein the synchronization operation instruction is an Lsync-type instruction.

4.   (Previously Presented) The apparatus of claim 2, wherein the synchronization operation instruction is an Lsync V,S-type instruction.

5.   (Previously Presented) The apparatus of claim 2, wherein, for a second synchronization operation instruction, a corresponding synchronization marker is inserted to in only the first queue.

6.   (Original) The apparatus of claim 5, wherein the second synchronization instruction is an Lsync-type instruction.

7-8.   (Canceled)

9.   (Previously Presented) The apparatus of claim 2, wherein the first queue includes two subqueues, including a first subqueue and a second subqueue, wherein the first subqueue is for holding the vector memory references and synchronization markers associated with the vector memory references and wherein the second subqueue is for holding a plurality of store data elements and synchronization markers associated with the store data elements, wherein each store data element in the second subqueue corresponds to a one of the memory requests in the first subqueue, and wherein the store data elements are loaded into the second subqueue decoupled from the loading of the memory requests into the first subqueue.

10.   (Previously Presented) The apparatus of claim 4, wherein the instruction-processing circuits include a data cache and wherein the Lsync V,S type instruction prevents subsequent scalar references from accessing the data cache until all vector references have been sent to an external cache and all vector writes have caused any necessary invalidations of the data cache.

11.   (Currently Amended) A method comprising:

providing a memory interface;

providing a plurality of queues connected to the memory interface, including a first queue and a second queue, <u>wherein</u> each of the ~~first~~ plurality of queues ~~for holding~~ <u>holds</u> ~~a plurality of~~ pending memory requests <u>and enforces an ordering in the commitment of the pending memory requests to memory</u>;

providing one or more instruction-processing circuits, wherein each instruction-processing circuit is operatively coupled through the plurality of queues to the memory interface;

inserting one or more memory requests into at least one of the queues based on a first memory operation instruction executed in one of the instruction-processing circuits;

inserting a first synchronization marker into the first queue and inserting a second synchronization marker into the second queue based on a synchronization operation instruction executed in one of the instruction-processing circuits; and

inserting one or more memory requests into at least one of the queues based on a second memory operation instruction executed in one of the instruction-processing circuits;

processing memory requests from the first queue; and

selectively halting further processing of memory requests from the first queue based on the first synchronization marker reaching a predetermined point in the first queue until the corresponding second synchronization marker reaches a predetermined point in the second queue;

wherein each of the memory requests is a memory reference.

12.   (Previously Presented) The method of claim 11, wherein the first queue stores only synchronization marks and vector memory references, and wherein the second queue stores only synchronization marks and scalar memory references.

13.   (Previously Presented) The method of claim 12, wherein the synchronization operation instruction is an Lsync-type instruction.

14.   (Previously Presented) The method of claim 12, wherein the synchronization operation instruction is an Lsync V,S-type instruction.

15-18. (Canceled)

19.   (Previously Presented) The method of claim 12, wherein the first queue includes two subqueues, including a first subqueue and a second subqueue, wherein the first subqueue stores the vector memory references and synchronization markers associated with the vector memory references and wherein the second subqueue stores a plurality of store data elements and synchronization markers associated with the store data elements, wherein each store data element in the second subqueue corresponds to one of the memory requests in the first subqueue, and wherein the store data elements are inserted into the second subqueue decoupled from the inserting of the memory requests into the first subqueue.

20.   (Previously Presented) The method of claim 14, wherein providing the instruction-processing circuits includes providing a data cache and wherein performing the Lsync V,S type instruction includes preventing subsequent scalar references from accessing the data cache until all vector references have been sent to an external cache and all vector writes have caused any necessary invalidations of the data cache.

21.   (Currently Amended) An apparatus comprising:
        a memory interface;
        a plurality of queues connected to the memory interface, including a first queue and a second queue, wherein each of the ~~first~~ plurality of queues ~~for holding~~ holds ~~a plurality of~~ pending memory requests and enforces an ordering in the commitment of the pending memory requests to memory;
        one or more instruction-processing circuits, wherein each instruction-processing circuit is operatively coupled through the plurality of queues to the memory interface and wherein each of the plurality of instruction-processing circuits includes:

AMENDMENT AND RESPONSE UNDER 37 CFR § 1.116 – EXPEDITED PROCEDURE   Page 6
Serial Number: 10/643,741   Dkt: 1376.733US1
Filing Date: August 18, 2003
Title: MULTISTREAM PROCESSING MEMORY-AND BARRIER-SYNCHRONIZATION METHOD AND APPARATUS

means for inserting one or more memory requests into at least one of the queues based on a first memory operation instruction executed in one of the instruction-processing circuits;

means for inserting a first synchronization marker into the first queue and inserting a second synchronization marker into the second queue based on a synchronization operation instruction executed in one of the instruction-processing circuits; and

means for inserting one or more memory requests into at least one of the queues based on a second memory operation instruction executed in one of the instruction-processing circuits;

means for processing memory requests from the first queue; and

means for selectively halting further processing of memory requests from the first queue based on the first synchronization marker reaching a predetermined point in the first queue until the corresponding second synchronization marker reaches a predetermined point in the second queue;

wherein each of the memory requests is a memory reference.

22.   (Previously Presented) The apparatus of claim 21, wherein means for inserting to the first queue operates for only vector memory requests and synchronizations, and means for inserting to the second queue operates for only scalar memory requests and synchronizations.

23.   (Previously Presented) The apparatus of claim 22, wherein the synchronization operation instruction is an Lsync-type instruction.

24.   (Currently Amended) A system comprising:

a plurality of processors, including a first processor and a second processor, wherein each of the processors includes:

a memory interface;

a plurality of Lsync queues connected to the memory interface, including a first Lsync queue and a second Lsync queue, <u>wherein</u> each of the plurality of Lsync queues

for holding holds a plurality of pending memory requests and enforces an ordering in the commitment of the pending memory requests to memory;

one or more instruction-processing circuits, wherein each instruction-processing circuit is operatively coupled through the plurality of Lsync queues to the memory interface and wherein each of the plurality of instruction-processing circuits inserts one or more memory requests into at least one of the Lsync queues based on a first memory operation instruction, inserts a first Lsync synchronization marker into the first Lsync queue and inserts a second Lsync synchronization marker into the second Lsync queue based on a synchronization operation instruction, and inserts one or more memory requests into at least one of the Lsync queues based on a second memory operation instruction; and

a Lsync synchronization circuit, operatively coupled to the plurality of Lsync queues, that selectively halts processing of further memory requests from the first Lsync queue based on the first Lsync synchronization marker reaching a predetermined point in the first Lsync queue until the corresponding second Lsync synchronization marker reaches a predetermined point in the second Lsync queue; and

one or more Msync circuits, wherein each of the Msync circuits is connected to the plurality of processors and wherein each of the Msync circuits includes:

a plurality of Msync queues, including a first Msync queue and a second Msync queue, each of the plurality of Msync queues for holding a plurality of pending memory requests received from the Lsync queues, wherein the first Msync queue stores only Msync synchronization markers and memory requests from the first processor, and the second Msync queue stores only Msync synchronization markers and memory requests from the second processor; and

an Msync synchronization circuit, operatively coupled to the plurality of Msync queues, that selectively halts further processing of the memory requests from the first Msync queue based on an Msync synchronization marker reaching a predetermined point in the first Msync queue until a corresponding Msync synchronization marker from the second processor reaches a predetermined point in the second Msync queue;

wherein each of the memory requests is a memory reference, wherein the memory reference is generated as a result of execution of instructions by instruction-processing circuits in each processor.

25.   (Previously Presented) The system of claim 24, wherein the Msync synchronization circuit includes a plurality of stall lines, wherein each of the stall lines is connected to one of the plurality of Msync queues and wherein each of the stall lines is for halting further processing of the memory requests from a corresponding Msync queue.

26.   (Previously Presented) The system of claim 24, wherein each processor includes a data cache and wherein each Msync synchronization circuit includes an external cache, wherein the data cache and the external cache are used to perform an Lsync V,S type instruction, wherein the Lsync V,S type instruction prevents subsequent scalar references from accessing the data cache until all vector references have been sent to the external cache in a corresponding Msync synchronization circuit and all vector writes have caused any necessary invalidations of the data cache.

27.   (Currently Amended) A method comprising:

providing a plurality of processors, including a first processor and a second processor, wherein each of the processors includes a memory interface, a plurality of Lsync queues connected to the memory interface, including a first Lsync queue and a second Lsync queue, wherein each of the plurality of Lsync queues holds pending memory requests and enforces an ordering in the commitment of the pending memory requests to memory, and one or more instruction-processing circuits, each of the instruction-processing circuits operatively coupled through the plurality of Lsync queues to the memory interface;

providing one or more Msync circuits, wherein each of the Msync circuits is connected to the plurality of processors and wherein each of the Msync circuits includes a plurality of Msync queues, including a first Msync queue and a second Msync queue, each of the plurality of Msync queues operatively coupled to the plurality of Lsync queues in one of the plurality of processors;

inserting one or more memory requests into at least one of the Lsync queues based on a first memory operation instruction executed in one of the instruction-processing circuits;

inserting a first Lsync synchronization marker into the first Lsync queue and inserting a second Lsync synchronization marker into the second Lsync queue based on a synchronization operation instruction executed in one of the instruction-processing circuits;

inserting one or more memory requests into at least one of the Lsync queues based on a second memory operation instruction executed in one of the instruction-processing circuits;

processing memory requests from the first Lsync queue;

selectively halting further processing of memory requests from the first Lsync queue based on the first Lsync synchronization marker reaching a predetermined point in the first Lsync queue until the corresponding second Lsync synchronization marker reaches a predetermined point in the second Lsync queue;

inserting Msync synchronization markers and memory requests received from the Lsync queues in the first processor into the first Msync queue;

inserting Msync synchronization markers and memory requests received from the Lsync queues in the second processor into the second Msync queue; and

selectively halting further processing of the memory requests from the first Msync queue based on an Msync synchronization marker reaching a predetermined point in the first Msync queue until a corresponding Msync synchronization marker from the second processor reaches a predetermined point in the second Msync queue;

wherein each of the memory requests is a memory reference.

28.   (Previously Presented) The method of claim 27, wherein selectively halting further processing of the memory requests from the first Msync queue includes sending a stall signal to the Msync queues.

29.   (Previously Presented) The method of claim 27, wherein selectively halting further processing of memory requests from the first Lsync queue includes performing an Lsync V,S type instruction, wherein performing the Lsync V,S type instruction includes preventing subsequent scalar references from accessing a data cache in the processor until all vector

references have been sent to an external cache in a corresponding Msync synchronization circuit

and all vector writes have caused any necessary invalidations of the data cache.